REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED
05-17-2001	Final	07-01-1996 to 06-30-97; May 2001

4. TITLE AND SUBTITLE

A Prototype Formal Methods Environment

5a. CONTRACT NUMBER

N00014-96-0364

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Teitelbaum, Ray(Tim)

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

GrammaTech, Inc.
317 N. Aurora Street
Ithaca, NY 14850

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

DCMAO
615 Erie Blvd West
Syracuse NY 13204-2408

10. SPONSOR/MONITOR'S ACRONYM(S)

ONR/DCMAO

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AILABILITY STATEMENT

No Restrictions

DISTRIBUTION STATEMENT A
Approved for Public Release
Distribution Unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The Synthesizer Generator is a system for generating language-based editors and interfaces from formal grammatical specifications. This project aimed to evaluate the utility of the Synthesizer Generator for building professional-quality user interfaces for formal-methods tools. As a test case, we used the Synthesizer Generator to prototype a new user-interface for the Cornell University's NuPRL theorem proving system, and delivered it to Cornell.

20010611 039

15. SUBJECT TERMS

User Interface, Formal Methods, Synthesizer Generator, NuPRL

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Ray(Tim) Teitelbaum
U	U	U		2	19b. TELEPHONE NUMBER *(Include area code)* 607-273-7340 Ext.25

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Summary

The Synthesizer Generator is a system for generating language-based editors and interfaces from formal grammatical specifications. This project aimed to evaluate the utility of the Synthesizer Generator for building professional-quality user interfaces for formal-methods tools. As a test case, we used the Synthesizer Generator to prototype a new user-interface for the Cornell University's NuPRL theorem proving system, and delivered it to Cornell.

The project had two primary co-objectives: (1) the development and delivery of a tool useful for NuPRL, and (2) prototyping generic facilities reusable for other applications. The prototype NuPRL editor addressed term editing and proof editing. We successfully demonstrated that the Synthesizer Generator could replicate the essential behavior of the handcrafted NuPRL user-interface. The distributed system structure prototyped in this project, in which the editor ran as a separate process and interacted with NuPRL by message passing, substantially influenced the architecture subsequently adopted by NuPRL.

The project began on 1 July 1996 and ran through 30 June 97. It followed completion of an ONR SBIR Phase I project entitled "User Interfaces for Rule-Based Formal Methods Environments", and contributed to the subsequent ONR SBIR Phase II project of the same name. The results of that SBIR project have since been commercialized as Ada-ASSURED for Windows and CodeSurfer®.

Results

Historically, most formal-methods systems have had minimal "glass teletype" command-line interfaces. Not surprisingly, formal-methods specialists have focused on their logics while providing the simplest, least-cost interface possible. Many environments have continued to ignore GUI interfaces and remain based on Emacs. For example, PVS uses standard Emacs (with a few minor embellishments) for text editing theory files and a shell window for interacting with the prover's command-line interface [1, 2].

The transcript produced by command-line interactions is just a linear sequence of inanimate "dead" characters; in contrast, *active documents* consist of "animate" interacting textual and graphical elements. Interaction via direct manipulation of active documents, one of the great interface revolutions of the 80's, has not been exploited by most formal methods systems. They have largely retained a temporal perspective (i.e., command sequence) and have not adopted the often more effective spatial perspective (i.e., active document). This has been true both at coarse granularity, e.g., theory browsing, and at fine granularity, e.g., term editing.

Fine-grained active documents. Effective editing of fine-grained active documents, e.g., language-sensitive term editing, is *not* easily provided by standard GUI elements. Rather, it must be laboriously programmed. The Synthesizer Generator, GrammaTech's commercial technology for generating language-sensitive editors and user-interfaces, is renowned for its editing support of fine-grained active documents.

Result 1. We continued development of a prototype NuPRL term editor implemented using the Synthesizer Generator, taking advantage of recent improvements in the Synthesizer Generator such as Motif GUI elements and a Scheme-based editor scripting language.

Proofs as documents. Most formal-methods systems do not view a proof as a hierarchical, editable, browsable document. They have adopted, instead, some form of goal-stack model in which the only persistent record of a proof is, at best, a linear proof script — a sequential record of the steps taken to make the proof [3, 4]. The process of trial-and-error proof is viewed as linear "time travel" forward and backward through the space of partial proof states rather than random access cut-and-paste in a partial proof document. In contrast to most formal-methods systems, NuPRL [5] has embraced the proof-as-document concept for years.

Result 2. We prototyped a proof-as-document-style proof editor for NuPRL.

Although options to prototype library browsers and efficient storage mechanisms for replicated fine-grained objects were not funded, some aspects of the intended work were eventually addressed under our subsequent SBIR Phase II project, and have now been commercialized in CodeSurfer®.

References

[1]. *The PVS Proof Checker: A Reference Manual (Draft)*. N. Shankar, S. Owre, and J. M. Rushby. Computer Science Laboratory, SRI International, Menlo Park, CA, March, 1993.

[2] *User Guide for the PVS Specification and Verification System*. S. Owre, N. Shankar, and J. M. Rushby. Computer Science Laboratory, SRI International, Menlo Park, CA, March, 1993.

[3] HOL: A proof generating system for higher-order logic. M. J. C. Gordon. In *VLSI Specifications, Verification and Synthesis*, pages 73-128. Kluwer, 1988.

[4] Constructions: A higher order proof system for mechanizing mathematics, T. Coquand and G. Huet. In *Lecture Notes in Computer Science*. Springer-Verlag, 1985. Volume 203.

[5] *Implementing Mathematics with the NuPRL Proof Development System*. R. L. Constable, et. al. Prentice-Hall, Englewood Cliff, N.J., 1986.